Graph Theory and Its Applications

Dr. G.H.J. Lanel

Lecture 3

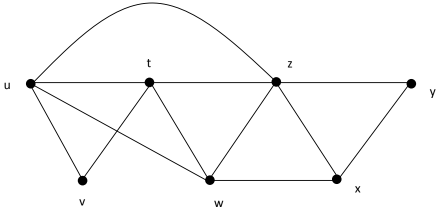
Outline

Outline

1. Connectivity
   - Generic Search
   - Depth First Search (DFS)
   - Breadth First Search (BFS)
   - Testing connectivity

Connected components

For any two vertices u and v in a graph G, if there is a walk from u to v, then G is connected.



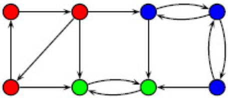
This is an equivalence relation (not for the digraphs) and hence leads to equivalence classes, which are called th connected components of the graph G.

The graph reduced to its connected components is acyclic (why ?)

Connected components

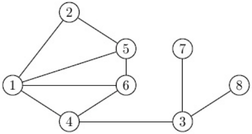
The notion of connectedness becomes more complex for digraphs, since the above equivalence relation no longer exist (it need to be symmetric).

In a directed graph $G = (V, E)$, u and v are strongly connected if there exists a walk from u to v and from v to u.



This is an equivalence relation.

$V(1) = \{2, 4, 5, 6\}$
$V(2) = \{1, 5\}$
$V(3) = \{4, 7, 8\}$
$V(4) = \{1, 3, 6\}$
$V(5) = \{1, 2, 6\}$
$V(6) = \{1, 4, 5\}$
$V(7) = \{3\}$
$V(8) = \{3\}$



Verify (strong) connectivity of a graph based on its adjacency list: start from vertex s, explore the graph, mark what you have visited.

Algorithm GenericSearch(G, s)

mark s, $L := \{s\}$

while $L \neq \phi$, **do**

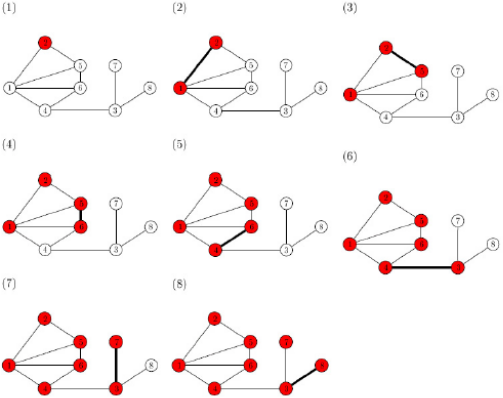
choose $u \epsilon L$,

if $\exists (u, v)$ such that v is unmarked **then** mark v, $L := L \cup \{v\}$,

else $L := L \backslash \{u\}$,

Below we marked the chosen vertices and the discovered vertices

L	*mark*
{2}	2
{2, 1}	1
{2, 1, 5}	5
{2, 1, 5, 6}	6
{1, 5, 6}	
{1, 5, 6, 4}	4
{5, 6, 4}	
{5, 4}	
{5, 4, 3}	3
{5, 3}	
{5, 3, 7}	7
{5, 3}	
{3}	
{3, 8}	8
{3}	
{}	



This algorithm has $2n$ steps : each vertex is added once and removed once. Its complexity is therefore linear in n.

Because of the choices, this algorithm allows for different versions. Let us use a LIFO list for L (Last In First Out) and choose for u the last element added to L. This is a depth first search (DFS).

Algorithm DepthFirstSearch(G, s)

mark s, $L := \{s\}$;

while $L \neq \phi$; **do**

u := last(L)

if $\exists(u, v)$ such that v is unmarked **then**

choose (u, v) with v of smallest index;

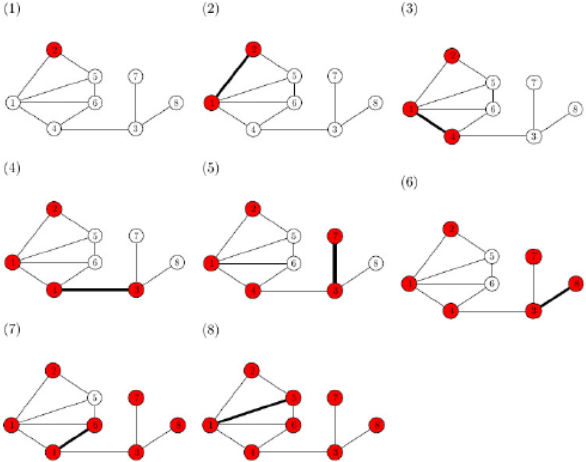
mark v; $L := L \cup \{v\}$;

else

$L := L \setminus \{u\}$

Below we marked the chosen vertices and the discovered vertices

L	*mark*
{2}	2
{2, 1}	1
{2, 1, 4}	4
{2, 1, 4, 3}	3
{2, 1, 4, 3, 7}	7
{2, 1, 4, 3}	
{2, 1, 4, 3, 8}	8
{2, 1, 4, 3}	
{2, 1, 4}	
{2, 1, 4, 6}	6
{2, 1, 4, 6, 5}	5
{2, 1, 4, 6}	
{2, 1, 4}	
{2, 1}	
{2}	
{}	



This algorithm builds longer paths than the generic one (depth first).

We now use a FIFO list for L (First In First Out) and choose for u the first element added to L. This is a breadth first search (BFS).

Algorithm BreadthFirstSearch(G, s)

mark s; $L := \{s\}$;

while $L \neq \phi$; **do**

u := first(L)

if $\exists(u, v)$ such that v is unmarked **then**

choose (u, v) with v of smallest index;

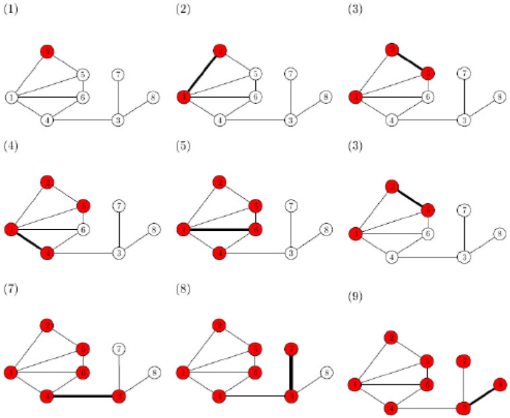
mark v; $L := L \cup \{v\}$;

else

$L := L \setminus \{u\}$

Below we marked the chosen vertices and the discovered vertices

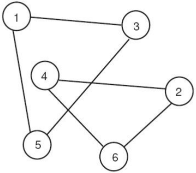
L	*mark*
{2}	2
{2, 1}	1
{2, 1, 5}	5
{1, 5}	
{1, 5, 4}	4
{1, 5, 4, 6}	6
{5, 4, 6}	
{4, 6}	
{4, 6, 3}	3
{6, 3}	
{3}	
{3, 7}	
{3, 7, 8}	8
{7, 8}	
{8}	
{}	



This algorithm builds a wider tree (breadth first).

The exploration algorithm finds the set of all vertices that can be reached by a path from a given vertex $u \in V$.

If the graph is undirected, each vertex in that set can follow a path back to u. They thus form the connected component $C(u)$ of u.



To find all connected components, repeat this exploration on a vertex of $V \setminus C(u)$, etc..

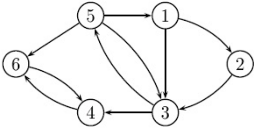
Proposition

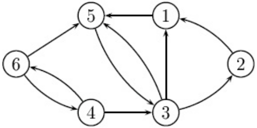
Let $G = (V, E)$ be a digraph and let $u \epsilon V$. If all $v \epsilon V$ there exists a path from u to v and a path from v to u, then G is strongly connected.

The exploration algorithm finds the set of all vertices that can be reached by a path from a given vertex $u \epsilon V$.

How can one find the vertices from which u can be reached?

Construct for that the inverse graph by reversing all arrows.



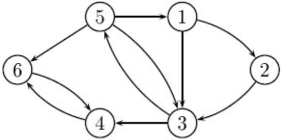


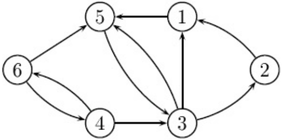
Show that the adjacency matrix of this graph is just A^T.

Proposition

Let $G = (V, E)$ be a digraph and let $u \epsilon V$. Let $R_+(u)$ be the vertices that can be reached from u and let $R_-(u)$ be the vertices that can reach u then the strongly connected component of u is $C(u) = R_+(u) \cap R_-(u)$

The exploration algorithm applied to the inverse graph, starting from u finds the set $R_-(u)$.





Here $R_+(v_6) = \{4, 6\}$ while $R_-(v_6) = \{4, 6, 5, 3, 1, 2\}$ hence $C(v_6) = \{4, 6\}$. Find the other strongly connected components.